**International Journal of Electronics,
Communication & Instrumentation Engineering
Research and Development (IJECIERD)
ISSN(P): 2249-684X; ISSN(E): 2249-7951
Vol. 5, Issue 4, Aug 2015, 21-28**

VERIFICATION OF AMBA-AHB BASED VERIFYING IP USING UVM METHODOLOGY

PALAKEETI NAVEEN KALYAN[1] & K JAYA SWAROOP[2]

[1] Student, Department, of ECE, QIS College of Engineering and Technology, Ongole, Andhra Pradesh, India

[2] Assistant Professor, Department, of ECE, Gandhiji Institute of Science and Technology, Jaggayyapet, Andhra Pradesh, India

ABSTRACT

This paper describes the verification of AMB- AHB based verifying IP using UVM (Universel Verification Methodology). AHB Is an Advanced High performance system Bus that supports multiple masters and multiple slaves. It Implements burst transfers, split transactions, single-cycle bus master handover, single-clock edge operation, wider data bus Configuration (64/128bits). Verification IP(Intellectual Property) is the one which provides a smart way to verify the AHB Components such as Master, Slave, Arbiter and Decoder. UVM is used for the verification of AHB Protocol which provides the best framework to achieve CDV (Coverage Driven Verification) which combines automatic test generation, self-checking test benches, and Coverage metrics to significantly reduce the time spent on verifying a design. An UVM test bench is composed of reusable Verification environements called VCs (verification Components). This paper examines the verification of VCs Which are structured to work with Verilog, VHDL, System Verilog and System C. AMBA Protocol based SoC it improves quality and reduces Schedule time it is the standard framework to build the verification environnement waveforms, code coverage is also discussed in the paper.

KEYWORDS: AMBA, AHB, CDV, UVM, TLM, VC, Test Bench, Sequencer

INTRODUCTION

AHB (Advanced High Performance Bus) is a new generation of AMBA (Advanced Microcontroller Bus Architecture) bus which is intended to address the requirements of high-performance synthesizable designs. AMBA AHB is a new level of bus which sits above the APB and implements the features required for high-performance, high clock frequency systems. UVM is a complete verification methodology that codifies the best practices for development of verification environments targeted at verifying large gate- count, IP-based SoCs (System on chip). Verification productivity stems from the ability to quickly develop individual verification components, encapsulate them into larger reusable verification components (VCs), and reuse them in different configurations and at different levels of abstraction. UVM uses a System Verilog implementation of standard Transaction Level Modeling (TLM) interfaces for modular communication between AHB components such as Master and Slave. The System Verilog UVM Class Library also provides various utilities to simplify the development and use of verification environments. These utilities support debugging by providing a user- controllable messaging utility. The System Verilog UVM Class Library provides global messaging facilities that can be used for failure reporting and general reporting purposes. The concept of coverage can get more complex, when we deal with the concept of functional coverage, cover groups and cover points. With the coverage

points, we can generate coverage report of your design and know the strength of your verification. Using UVM has test case multiple tests are compiled together and a test case can be chosen at the run time using command line the constraint random test cases in the UVM are much shorter than the directed test cases with UVM test class we don't need to recompile the environment again to run a new test case which saves huge time for big verification environments. In this paper, how to put up the simulation environment using UVM Methodology is introduced. The simulation environment using by UVM is efficient and high-performance. The paper discusses the usage experience to verify a SoC by the IP and the AHB during verify the SoC, a great deal of visual simulation waveform inspection is required. To generate the waveform will be time-consuming. The paper presents SoC bus transaction verification IP written by Open Verification Methodology, which are called reference model to compare the behaviour of the DUT. The IP can be integrated into UVM simulation environment. We can greatly reduce the waveform inspection time.

AMBA-AHB Design

The AHB bus will have the four components namely master, slave, arbiter, decoder. As referring to the figure 1 AHB block diagram, the signals HBUSREQx, HTRANS, HWDATA, HSIZE, HLOCK, HADDR are the master signals and the HRDATA, HRESP, HREADY are the slave signals. Starting from HBUSREQx the master will send a request signal to the arbiter (i.e to the HGRANT signals) to get the granted access to the bus. Then the arbiter indicates when the master will granted uses the bus. After getting the access then the AHB transfer can commence, and the AHB bus also has provisions of several masters but only one master must be access for the data transfer and that master only can access the slaves at a given time. A granted bus master starts an AMBA AHB transfer by driving the address and control signals. Here the data can transfer in the form of four ways as idle, busy, non-sequential, Sequential. The bus master will uses the HTRANS signal to what type of data is going to be transfer and by using HWDATA signal the bus master can transfer the data from master to the slave.In the figure 2 Only one master can access the slaves at the given time. All of the other masters must respect this and wait until the bus is assigned to them, and the slave will use the HRDATA signal to send the response to the master signal and the slave can drive HREADY low to stretch the length of a bus cycle. The master must observe the status of the HRESP. If RETRY or SPLIT is given, the master must immediately drive the IDLE value on its HTRANS output. When the HLOCK is asserted high indicates that the master requires locked access to the bus and no other master should be granted the bus until this signal is LOW. The main feature of the AHB is that. In fact, the address phase of any transfer occurs during the data phase of the previous transfer. This overlapping of address and data is fundamental to the pipelined nature of the bus and allows for high performance operation.

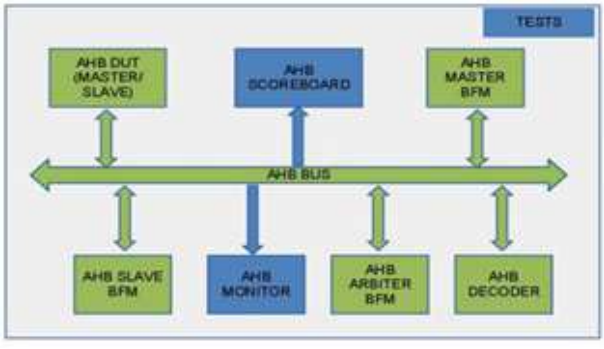

Figure 1 : AHB Bus Structure

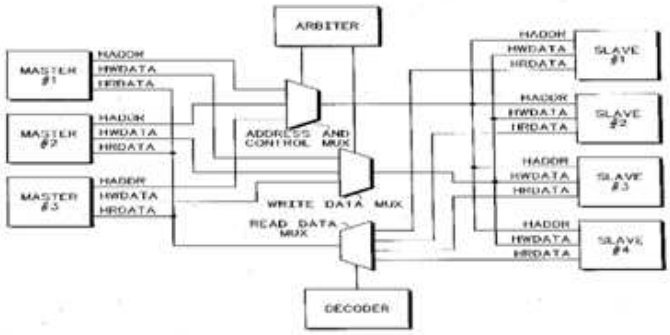

Figure 2: AHB Design

Architecture of UVM Test Bench

An UVM test bench is composed of reusable verification environments called Verification Components (VCs). The VCs are applied to the device under test (DUT) to verify the implementation of the AHB protocol. Architecture of UVMTest Bench is shown in Figure 3 Cadence design system and mentor graphics as a joint effort to provide a common methodology to verification .By using of UVM we can write number of test cases,independent of test bench environment. As shown in the figure 4 executing flow in the UVM hierarchy. It consists of UVM components, UVM test. UVM component classes are sequencer, driver, monitor, scoreboard.

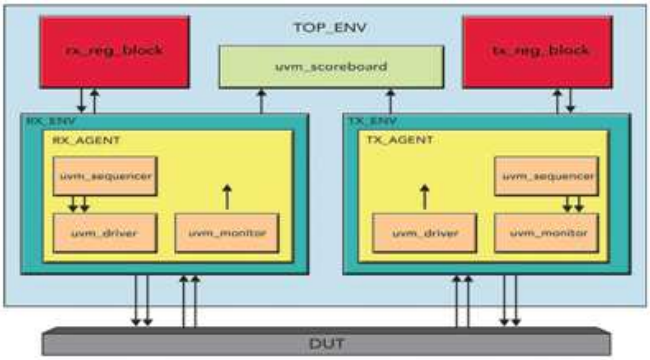

Figure 3: UVM Architecture

The three main building blocks of a test bench in UVM based verification are

UVM_env: It is the top level component of the verification components.uvm_envisextendedfrom uvm_component. It is used to create and connect the uvm_components like drivers, monitors, sequencers.It can also used as sub environment in another environment.

UVM test: The uvm_test class defines the test scenario for the test bench specified in the test. The test class enables configuration of the test bench and verification components.

UVM Verification Components: Sequencer (stimulus generator): The sequencer generates stimulus data and passes it to a driver for execution. The uvm_sequencer is the base class of uvm class library contains all of the base functionality required to allow a sequence to communicate with a driver.

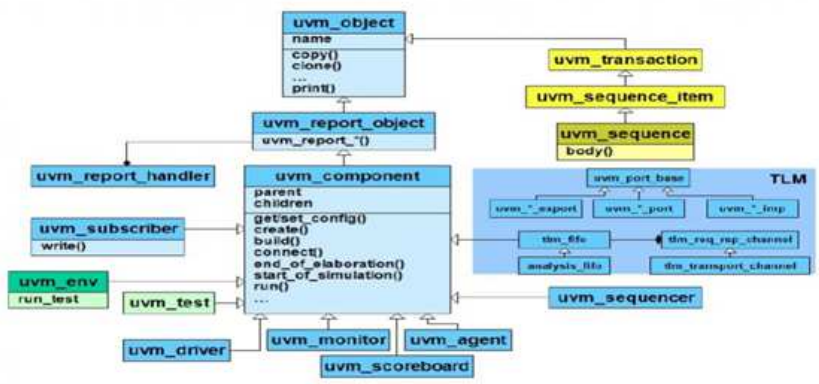

Figure 4: UVM Hierachy

UVM_object: Is the virtual base class for all components and transactions in an UVM environment.

UVM_sequencer: It is a verification component that mediates the generation and flow of data between sequences and a driver. The sequencer has a collection of sequences associated with it called a sequence library. The collection of sequences used by a sequencer is called sequence library. This type of component is also referred to as a driver sequencer.

UVM_generator: It is used to generate the test vectors.

UVM-driver: generator create inputs at high level of abstraction namely as transactions like read write operation. The driver convert this input signal into actual design inputs , as defined in the specification of the design interfaces. A driver is a verification component that connects at the pin-level interface to the DUT. It contains one or more transaction-level interfaces to communicate with other transaction-level components in the verification environment.

UVM_monitor: Monitor reports the protocol violation and identifies all the transactions. Monitors are two types, Passive and active. Passive monitors do not drive any signals.Active monitors can drive the DUT signals. Sometimes this is also referred as receiver.Monitor converts the state of the design and its outputs to a transaction abstraction level so it can be stored in a 'score-boards' database to be checked later on.Monitor converts the pin level activities in to high level.

UVM_agent: A device that contains the standard components necessary to drive HDL signals (the driver), provide stimulus to the driver (the sequencer) and collect data items along with enforcing checks and tabulating coverage (the monitor). An agent is capable of independent operation. An Agent has two operating modes

Active mode: In this mode, the agent emulates a device in the system and drives DUT signals. This mode requires that the agent instantiate a driver and sequencer. A monitor also is instantiated for checking and coverage.

Passive mode: In this mode, the agent does not instantiate a driver or sequencer and operates passively. Only the monitor is instantiated and configured. This mode is used only when checking.

UVM_scoreboard:A scoreboard predicts the response of the design dynamically. The stimulus applied to the design is provided to a transfer function. It is the transfer function that performs all transformation operations on the stimulus in order to produce the form of the final response then inserts it in a data structure. Score board will have two analysis ports. One is used for getting packets from the driver and other from the receiver

Transaction: it means basically what needs to be tested. Examples of transaction are instruction, data, parameters, where instruction represents the high-level tasks to be executed, such as a READ,WRITE,NO-OP,LOAD,etc. The data represents such as address, data, number of cycles, etc, and the parameters represents a mode, a size,etc.

Interface: interface is the mechanism to connect Testbench to the DUT just named as bundle of wires(e.g. connecting two hardware unit blocks with the help of physical wires)

The components in an UVM verification environment are derived either directly or indirectly from the uvm_component classes

- Build() : This phase is used to construct various child components/ports/exports and configures them.
- Connect() : This phase is used for connecting the ports/exports of the components.
- End of elaboration() : This phase is used for using the components if required.
- Start of simulation() : This phase is used to print the banners and topology.
- Run() : In this phase, main body of the test is executed where all threads are forked off.

RESULTS AND DISCUSSIONS

The results of verification components such as Master Agent and the Slave Agent of the UVM Environment are presented. According to Test Plan, the test cases are verified by developing the Verification IP for the AHB Protocol. The Test Cases are written in the form of sequences in the Sequencer using System Verilog. The sequencer drives the sequences to the driver and There by to Score Board. In the scoreboard, the actual output is compared with the expected one. If the obtained output matches with the expected result then we conclude that the verification iscompleted successfully. By using the tool Questa, the Verification of AHB Components known as VC's such as Master Agent and Slave agent are done and the log files for the test cases are generated and simulated waveforms are achieved and shown in Figures 5 & 6

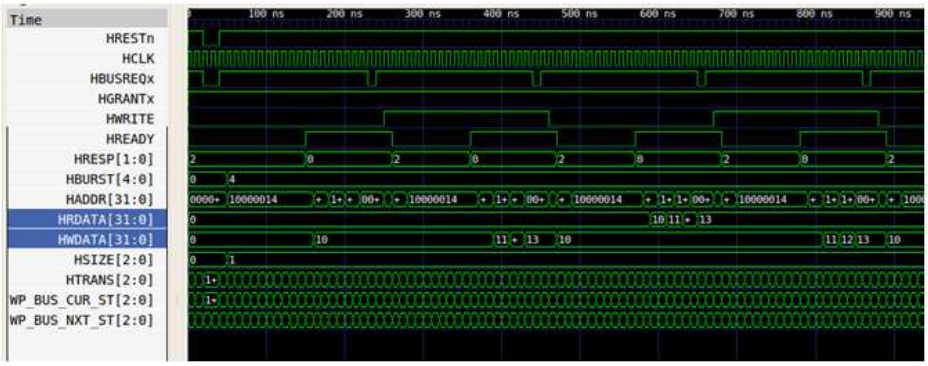

Figure 5: Read Transaction Wrapburst

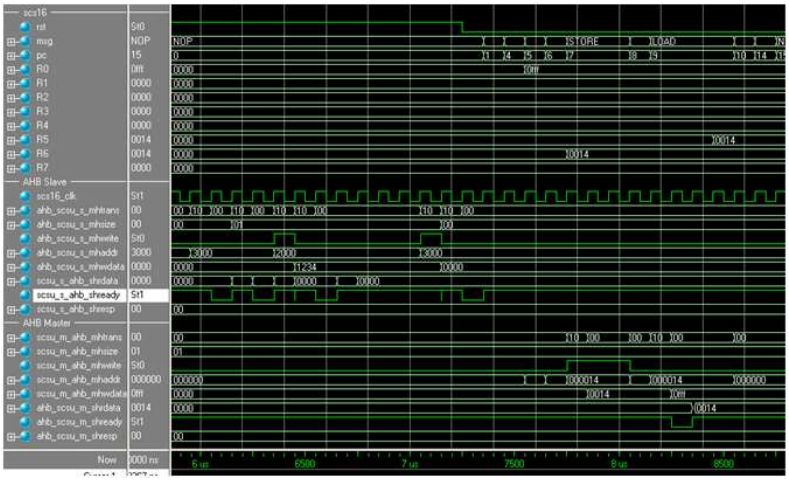

Figure 6: Read and Write Transaction for Wrapburst 4

CONCLUSIONS

Verification of AHB Protocol for Single Master-Single Slave has been verified by developing the Verification IP using the UVM methodology. UVM used System Verilog implementation of standard TLM interfaces for modular communication between Verification Components. Coding of AHB components such as Master Agent and Slave Agent has been done in System Verilog using UVM. The Verification IP s developed using tool Questa and it is sure that developed VIP is also compatible with other Tools such as Cadence,Synopsys etc. The developed Verification IP can be used in the verification of SoCs.

ACKNOWLEDGEMENTS

We the authors of this paper would like to greatfully acknowledge with thank K Jaya Swaroop for explaining the changes made in the specifications we are grateful for many insightful comments about our specifications that helped us improve our paper. .forconstant encouragement and support for enabling us to submit this paper.

REFERENCES

1. ARM Ltd. AMBA Specification Rev.2.0 May 1999.http://www.arm.com
2. Open Verification Methodology: Fulfilling the Promise of System Verilog' by Thomas L. Anderson, Product Marketing Director Cadence Design Systems, Inc.
3. System Verilog for Verification: A Guide to Learning the Test bench Language Features'byChris Spears.l.: Springer, 2006
4. Chrisspear, system Verilog for verification, New York: springer, 2006. Rath A.W, Esen.V and Ecker.W , "A transaction –oriented UVM-based library for verification of analog behavior " Publication Year: 2014 Page(s): 806 – 811
5. Stuart Sutherland, Don Mills, "Synthesizing System Verilog Busting the myth that System Verilog is only for Verification"
6. Soo-Yun Hwang and Kyoung-Sun Jhang, "An Improved Implementation Method Of AHB BusMatrix" SOC

Conference 2005, Proceedings, IEEE International pp. 211-214

7. Mulani, " Level Verification Using SystemVerilog " Emerging Trends in Engineering and Technology (ICETET), 2009 2nd International Conference on 16-18 Dec.2009 pp.378-380
8. Pockrandt, M. Herber, P and Glesner, S, " Model checking a SystemC/TLM design of the AMBA AHB Protocol " Embedded Systems for Real-Time Multimedia (ESTIMedia), 2011 9th IEEE Symposium on 13-14 Oct.2011 pp.66 – 75
9. IEEE Draft Standard for System Verilog - Unified Hardware Design,Specification and Verification Language, IEEE P1800/DS, February2012 pp.1-1304